Introducción a la Bioinformática:

Hidden Markov Models (HMMs)

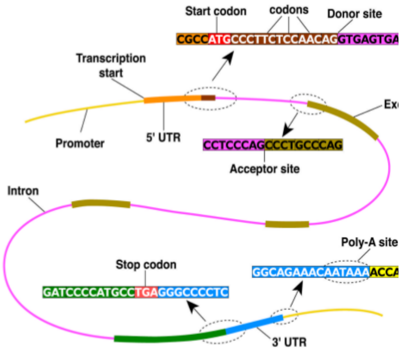
Luis Garreta

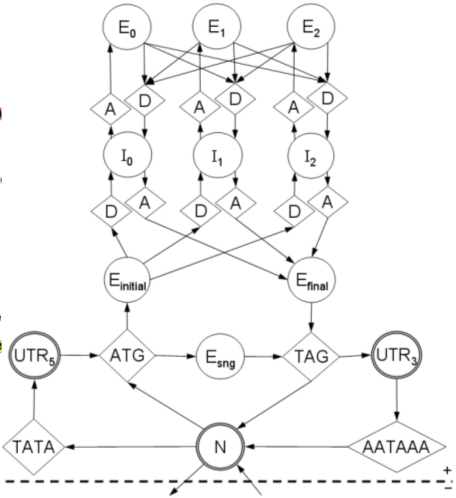
Doctorado en Ingeniería
Pontificia Universidad Javeriana – Cali

April 23, 2018

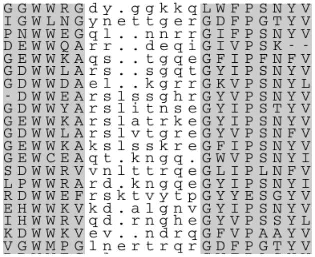
Applications of HMMs

Gene Finding and Prediction

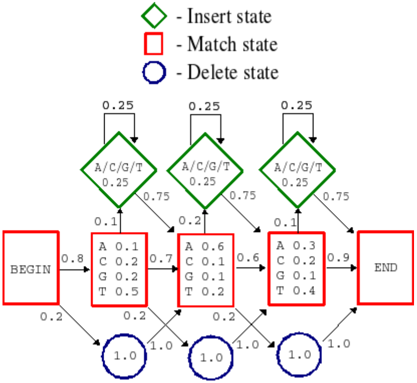




Multialignments and Profiles



```
GGWWRGdy.ggkkqLWFPSNYV
IGWLNGynettgerGDFPGTYV
PNWWEGql..nnrrGIFPSNYV
DEWWQArr..deqiGIVPSK--
GEWWKAqs..tgqeGFIPFNFV
GDWWLArs..sgqtGYIPSNYV
GDWWDAel..kgrrGKVPSNYL
-DWWEArslssghrGYVPSNYV
GDWWYArslitnseGYIPSTYV
GEWWKArslatrkeGYIPSNYV
GDWWLArslvtgreGYVPSNFV
GEWWKAkslsskreGFIPSNYV
GEWCEAqt.kngq.GWVPSNYI
SDWWRVvnlttrqeGLIPLNFV
LPWWRArd.kngqeGYIPSNYI
RDWWEFrsktvytpGYYESGYV
EHWWKVkd.algnvGYIPSNYV
IHWWRVqd.rngheGYVPSSYL
KDWWKVev..ndrqGFVPAAYV
VGWMPGlnertrqrGDFPGTYV
```



Other Applications of HMMs

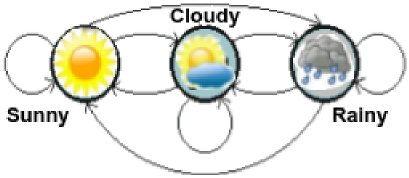
- Speech recognition
- Optical character recognition
- Spell checking

Architecture of HMMs

Markov Chains:

Markov Assumption

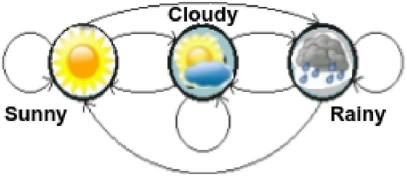
- Three states: Sunny, Cloudy, and Rainy



- Markov Assumption: The state of the model depends only upon the previous states of the model
- Order n Model (First Order): The choice of state is made purely on the basis of the previous state

Markov Chains:

State Transition Matrix (A)



A= (rows: YESTERDAY; columns: TODAY)

YESTERDAY \ TODAY	Sunny	Cloudy	Rainy
Sunny	0.5	0.25	0.25
Cloudy	0.375	0.125	0.375
Rainy	0.125	0.625	0.375

If it was sunny yesterday, there is a probability of 0.5 that it will be sunny today, and 0.25 that it will be cloudy or rainy.

Markov Chains:
Vector of Initial Probabilities (π)

$\pi =$

Sunny	Cloudy	Rainy
1.0	0.0	0.0

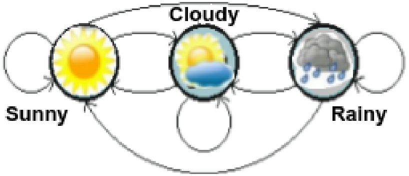
- To initialize such a system, we need to state what the weather was (or probably was) on the day after creation;
- So, we know it was sunny on day 1

Markov Chains:

First Order Markov Process

- **States:** Three states: sunny, cloudy, rainy
- π **vector:** Probability of the system in each states at time 0
- **State transition Matrix:** Probability of the weather given the previous day's weather

Any system that can be described in this manner is a Markov process.



$\pi =$

Sunny	Cloudy	Rainy
1.0	0.0	0.0

A=

	Sunny	Cloudy	Rainy
Sunny	0.5	0.25	0.25
Cloudy	0.375	0.125	0.375
Rainy	0.125	0.625	0.375

Hidden Markov Models

- In some cases the patterns that we wish to find are not described sufficiently by a Markov process.
- A hermit for instance may not have access to direct weather observations, but does have a piece of seaweed.

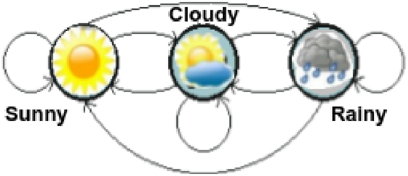
- Sea and weather lore: seaweeds are weather predictors (they absorb atmospheric moisture)
- The seaweed is probabilistically related to the state of the weather:

Hidden Markov Models: Two sets of States

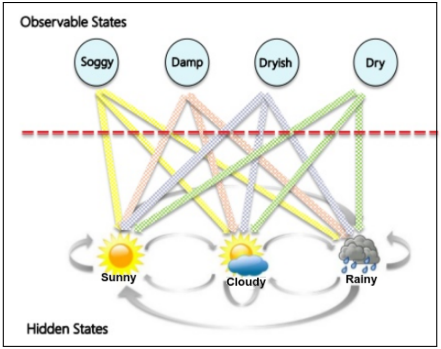
In this case we have two sets of states:

- observable states (the state of the seaweed) and
- hidden states (the state of the weather).

We wish to devise an algorithm for the hermit to forecast weather from the seaweed and the Markov assumption without actually ever seeing the weather.



Hidden Markov Models: Hidden and Observable States

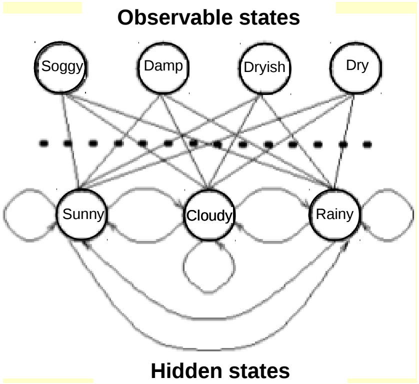


- Hidden states (the true weather) are modeled by a simple Markov process.
- So, they are all connected to each other.
- The new connections represent: *the probability of generating a particular observed state given that the Markov process is in a particular hidden state.*

Hidden Markov Models: Emission Matrix

The probabilities of the observable states given a particular hidden state:

	Seaweed			
Weather	Dry	Dryish	Damp	Soggy
A= Sunny	0.60	0.20	0.15	0.05
Cloudy	0.25	0.25	0.25	0.25
Rainy	0.05	0.10	0.35	0.50

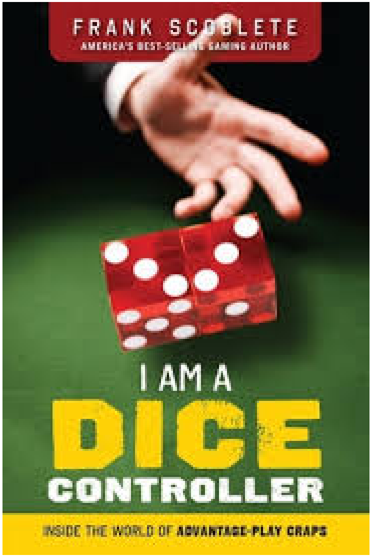


All probabilities "entering" an observable state will sum to 1 :
$Pr(Obs|Sunny) + Pr(Obs|Cloudy) + Pr(Obs|Rainy) = 1$

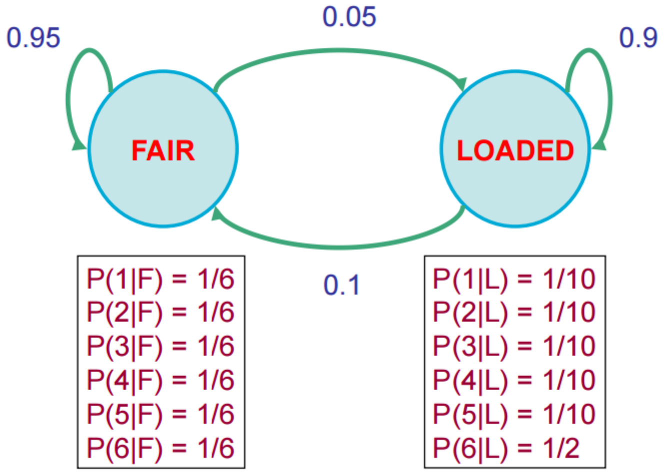
Example 2: The Dishonest Casino HMM

Example: The Dishonest Casino

- Game:
  1. You bet \$1
  2. You roll
  3. Casino player rolls
  4. Highest number wins \$2
- The casino has two dice:
  - ✓ Fair die:
    P(1)=P(2)=P(3)=P(4)=P(5)=P(6)
  - ✓ Loaded die:
    P(1)=P(2)=P(3)=P(4)=P(5)
    P(6)=1/2
- Casino player switches between fair and loaded die (not too foten, and not for too long)


FRANK SCOBLETE
AMERICA'S BEST-SELLING GAMING AUTHOR
I AM A
DICE
CONTROLLER
INSIDE THE WORLD OF ADVANTAGE-PLAY CRAPS

The dishonest casino model



Question # 1 – Evaluation

GIVEN:

A sequence of rolls by the casino player

$$\underbrace{1245526462146146136136661664661636616366163616515615115146123562344}_{\boxed{\text{Prob} = 1.3 \times 10^{-35}}}$$

QUESTION:

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question # 2 – Decoding

GIVEN:

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616		515615115146123562344

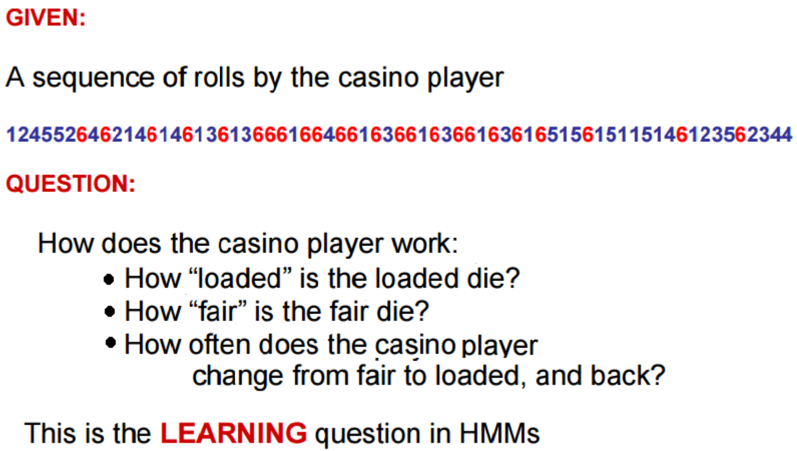
FAIR	LOADED	FAIR

QUESTION:

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

Question # 3 – Learning



GIVEN:

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

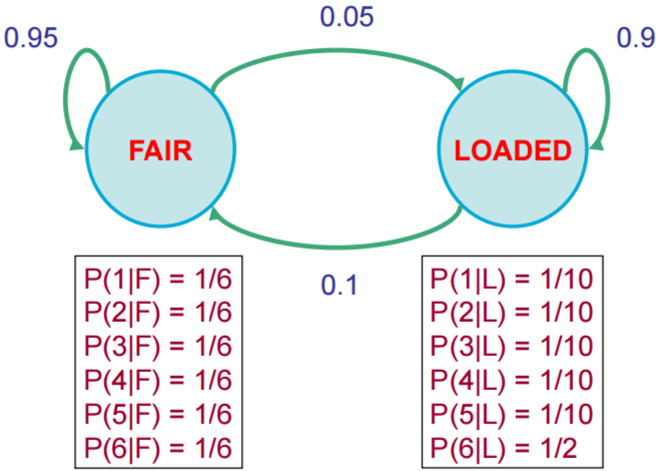
QUESTION:

How does the casino player work:

- How "loaded" is the loaded die?
- How "fair" is the fair die?
- How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

The dishonest casino model



Definition of a hidden Markov model

- Alphabet $\Sigma = \{ b_1, b_2, \ldots, b_M \}$
- Set of states $Q = \{ 1, \ldots, K \}$ $(K = |Q|)$
- Transition probabilities between any two states

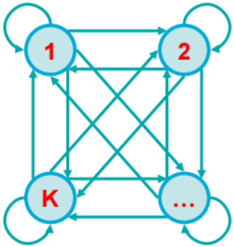
  a_{ij} = transition probability from state i to state j

  $a_{i1} + \ldots + a_{iK} = 1$, for all states i
- Initial probabilities a_{0i}

  $a_{01} + \ldots + a_{0K} = 1$
- Emission probabilities within each state

  $e_k(b) = P(x_i = b \mid \pi_i = k)$

  $e_k(b_1) + \ldots + e_k(b_M) = 1$



Hidden states and observed sequence

At time step t,

π_t denotes the (hidden) state in the Markov chain

x_t denotes the symbol emitted in state π_t

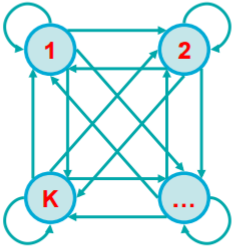
A path of length N is: $\pi_1, \pi_2, \ldots, \pi_N$

An observed sequence of length N is: $x_1, x_2, \ldots, x_N$

An HMM is "memory-less"

At time step t, the only thing that affects the next state is the current State, π_t

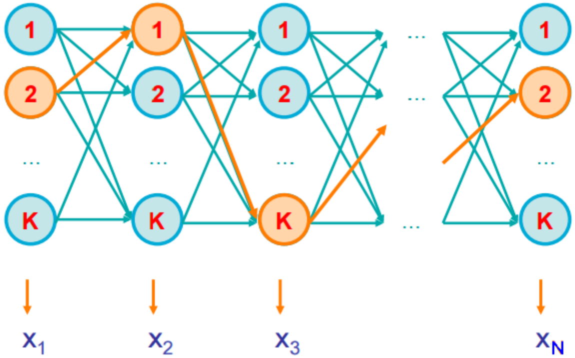
$$P(\pi_{t+1} = k \mid \text{"whatever happened so far"})$$
$$= P(\pi_{t+1} = k \mid \pi_1, \pi_2, \ldots, \pi_t, x_1, x_2, \ldots, x_t)$$
$$= P(\pi_{t+1} = k \mid \pi_t)$$



A parse of a sequence

Given a sequence $x = x_1 \ldots\ldots x_N$,

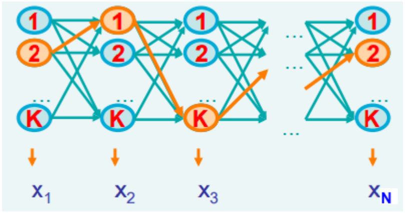
A parse of x is a sequence of states $\pi = \pi_1, \ldots\ldots, \pi_N$



Likelihood of a parse: $P(x, \pi)$

Given a sequence $x = x_1 \ldots\ldots, x_N$
and a parse $\pi = \pi_1, \ldots\ldots, \pi_N$,

How likely is the parse
(given our HMM)?



$$P(x, \pi) = P(x_1, \ldots, x_N, \pi_1, \ldots\ldots, \pi_N)$$

$$= P(x_N, \pi_N \mid x_1 \ldots x_{N-1}, \pi_1, \ldots\ldots, \pi_{N-1})\, P(x_1 \ldots x_{N-1}, \pi_1, \ldots\ldots, \pi_{N-1})$$

$$= P(x_N, \pi_N \mid \pi_{N-1})\, P(x_1 \ldots x_{N-1}, \pi_1, \ldots\ldots, \pi_{N-1})$$

$$= \ldots$$

$$= P(x_N, \pi_N \mid \pi_{N-1})\, P(x_{N-1}, \pi_{N-1} \mid \pi_{N-2}) \ldots\ldots P(x_2, \pi_2 \mid \pi_1)\, P(x_1, \pi_1)$$

$$= P(x_N \mid \pi_N)\, P(\pi_N \mid \pi_{N-1}) \ldots\ldots P(x_2 \mid \pi_2)\, P(\pi_2 \mid \pi_1)\, P(x_1 \mid \pi_1)\, P(\pi_1)$$

$$= a_{0\pi 1}\, a_{\pi 1 \pi 2} \ldots\ldots a_{\pi N-1 \pi N}\, e_{\pi 1}(x_1) \ldots\ldots e_{\pi N}(x_N)$$

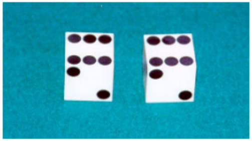
$$= \prod_{i=1}^{N} a_{\pi_{i-1}\pi_i} e_{\pi_i}(x_i)$$

Example: the dishonest casino $P(x, \pi)$, $\pi = FFFFFF \ldots FF$

What is the probability of a sequence of rolls

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

and the parse



π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

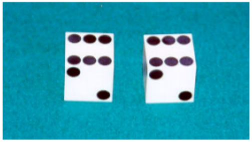
(say initial probs $a_{0,\text{Fair}} = ½$, $a_{0,\text{Loaded}} = ½$)

½ × P(1 | Fair) P(Fair | Fair) P(2 | Fair) P(Fair | Fair) … P(4 | Fair) =

$½ \times (1/6)^{10} \times (0.95)^9 = 5.2 \times 10^{-9}$

Example: the dishonest casino $P(x, \pi)$, $\pi = LLLLLL...LL$

So, the likelihood the die is fair in all this run is 5.2×10^{-9}



What about

π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

½ × P(1 | Loaded) P(Loaded|Loaded) … P(4 | Loaded) =

$½ \times (1/10)^8 \times (1/2)^2 (0.9)^9 = 4.8 \times 10^{-10}$

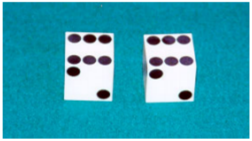
Therefore, it is more likely that the die is fair all the way, than loaded all the way

Example: the dishonest casino, loglikehood-ratio

A likelihood ratio test is a statistical test used for comparing the goodness of fit of two models, one of which (the null model) is a special case of the other (the alternative model)

$$log(\frac{P(X|\pi_{Fair})}{P(X|\pi_{Loaded})}) = log(\frac{5.2-09}{4.8e-10}) = 10.76$$

Example: the dishonest casino: Suspicion of loaded dice



Let the sequence of rolls be:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

And let's consider π = F, F,…, F

$P(x, \pi) = ½ \times (1/6)^{10} \times (0.95)^9 = 5.2 \times 10^{-9}$
(same as before)

And for π = L, L,…, L:

$P(x, \pi) = ½ \times (1/10)^4 \times (1/2)^6 (0.9)^9 = 3.02 \times 10^{-7}$

So, the observed sequence is ~100 times more likely if a loaded die is used

Clarification of notation

P[x | M]: The probability that sequence x was generated by the model

The model is: architecture (#states, etc) + parameters $\theta = a_{ij}, e_i(.)$

So, P[x | M] is the same as P[x | θ], and P[x], when the architecture, and the parameters, respectively, are implied

Similarly, P[x, π | M], P[x, π | θ] and P[x, π] are the same when the architecture, and the parameters, are implied

In the LEARNING problem we write P[x | θ] to emphasize that we are seeking the θ* that maximizes P[x | θ]

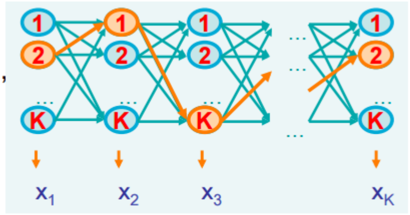
HMM Problems

What we know

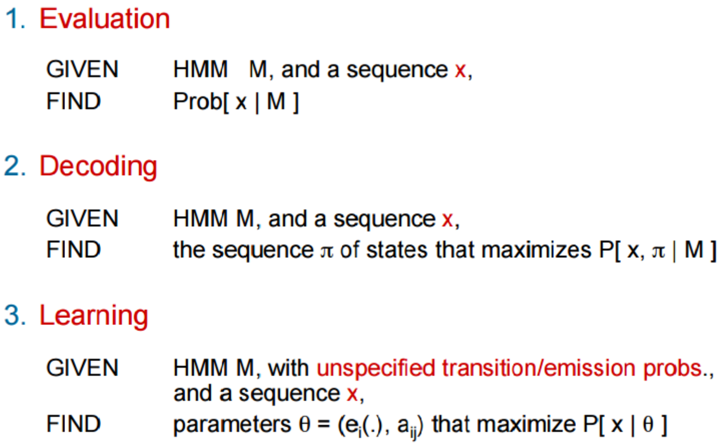
Given a sequence $x = x_1,\ldots,x_N$
and a parse $\pi = \pi_1,\ldots,\pi_N$,

we know how to compute
how likely the parse is:

$P(x, \pi)$



What we would know



1. Evaluation

GIVEN	HMM M, and a sequence x,
FIND	Prob[x | M]

2. Decoding

GIVEN	HMM M, and a sequence x,
FIND	the sequence π of states that maximizes P[x, π | M]

3. Learning

GIVEN	HMM M, with unspecified transition/emission probs., and a sequence x,
FIND	parameters $\theta = (e_i(.), a_{ij})$ that maximize P[x | θ]

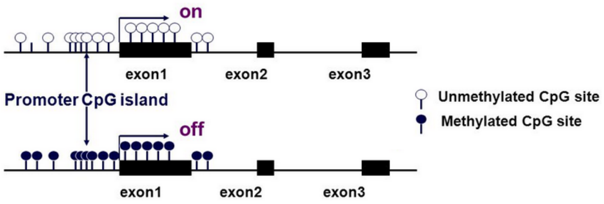
Problem 2: Decoding

Find the best parse of a sequence

CpG Islands Example

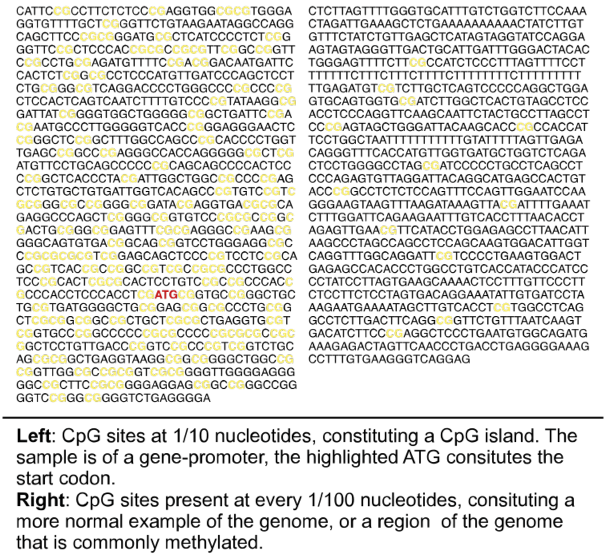
- Region of the genome with high frequency of CpG sites than the rest of the genome
- Formally: CpG island is a region with al least 200bp, and GC percentage that is greater than 50%
- Approximately 15% of the CpG sites are found in CpG islands in the promoter regions of some 70% of protein-coding genes[a]

GpC islands appear to be related with coding DNA.



[a]Sandelin A, Carninci P, Lenhard B, Ponjavic J, Hayashizaki Y, Hume DA Nat Rev Genet. 2007 Jun; 8(6):424-36.

CPG Islands Sequences



```
CATTCCGCCTTCTCTCCCGAGGTGGCGCGTGGGA
GGTGTTTTGCTCGGGTTCTGTAAGAATAGGCCAGG
CAGCTTCCCGCGGGATGCGCTCATCCCCTCTCGG
GGTTCCGCTCCCACCGCGCCGCGTTCGGCCGGTT
CCGCCTGCGAGATGTTTTCCGACGGACAATGATTC
CACTCTCGGCGCCTCCCATGTTGATCCCAGCTCCT
CTGCGGGCGTCAGGACCCCTGGGCCCCGCCCCG
CTCCACTCAGTCAATCTTTTGTCCCCGTATAAGGCG
GATTATCGGGGTGGCTGGGGGCGGCTGATTCCGA
CGAATGCCCTTGGGGGTCACCCGGGAGGGAACTC
CGGGCTCCGGCTTTGGCCAGCCCGCACCCCTGGT
TGAGCCGGCCCGAGGGCCACCAGGGGGCGCTCG
ATGTTCCTGCAGCCCCCCGCAGCAGCCCCACTCC
CCGGCTCACCCTACGATTGGCTGGCCGCCCCGAG
CTCTGTGCTGTGATTGGTCACAGCCCGTGTCCGTC
GCGGGCGCCGGGGCGGATACGAGGTGACGCGCA
GAGGCCCAGCTCGGGGCGGTGTCCCGCGCCGGC
GACTGCGGGCGGAGTTTCGCGAGGGCCGAAGCG
GGGCAGTGTGACGGCAGCGGTCCTGGGAGGCGC
CCGCGCGCGTCGGAGCAGCTCCCCGTCCTCCGCA
GCCGTCACCGCCGGCCGTCGCCGCGCCCTGGCC
TCCCGCACTCGCGCACTCCTGTCCGCCGCCCACC
GCCCACCTCCCACCTCGATGCGGTGCCGGGCTGC
TGCGTGATGGGGCTGCGGAGCGGCGCCCTGCGG
CTCGCGGCGGCCGCTGCTCGCGCTGAGGTGCGT
CGGTGCCCGGCCCCCCGCGCCCCCGCGCGCCGC
GGCTCCTGTTGACCCGGTCCGCCCGTCGGTCTGC
AGCGCGGCTGAGGTAAGGCGGCGGGGCTGGCCG
CGGTTGGCGCCGCGGTCGCGGGGTTGGGGAGGG
GGCCGCTTCCGCGGGGAGGAGCGGCCGGGCCGG
GGTCCGGGCGGGGTCTGAGGGGA
```

```
CTCTTAGTTTTGGGTGCATTTGTCTGGTCTTCCAAA
CTAGATTGAAAGCTCTGAAAAAAAAAACTATCTTGT
GTTTCTATCTGTTGAGCTCATAGTAGGTATCCAGGA
AGTAGTAGGGTTGACTGCATTGATTTGGGACTACAC
TGGGAGTTTTCTTCGCCATCTCCCTTTAGTTTTCCT
TTTTTTCTTTCTTTCTTTTCTTTTTTTTCTTTTTTTTT
TTGAGATGTCGTCTTGCTCAGTCCCCCAGGCTGGA
GTGCAGTGGTGCGATCTTGGCTCACTGTAGCCTCC
ACCTCCCAGGTTCAAGCAATTCTACTGCCTTAGCCT
CCCGAGTAGCTGGGATTACAAGCACCCGCCACCAT
TCCTGGCTAATTTTTTTTTTTTGTATTTTTAGTTGAGA
CAGGGTTTCACCATGTTGGTGATGCTGGTCTCAGA
CTCCTGGGGCCTAGCGATCCCCCTGCCTCAGCCT
CCCAGAGTGTTAGGATTACAGGCATGAGCCACTGT
ACCCGGCCTCTCTCCAGTTTCCAGTTGGAATCCAA
GGGAAGTAAGTTTAAGATAAAGTTACGATTTTGAAAT
CTTTGGATTCAGAAGAATTTGTCACCTTTAACACCT
AGAGTTGAACGTTCATACCTGGAGAGCCTTAACATT
AAGCCCTAGCCAGCCTCCAGCAAGTGGACATTGGT
CAGGTTTGGCAGGATTCGTCCCCTGAAGTGGACT
GAGAGCCACACCCTGGCCTGTCACCATACCCATCC
CCTATCCTTAGTGAAGCAAAACTCCTTTGTTCCCTT
CTCCTTCTCCTAGTGACAGGAAATATTGTGATCCTA
AAGAATGAAAATAGCTTGTCACCTCGTGGCCTCAG
GCCTCTTGACTTCAGGCGGTTCTGTTTAATCAAGT
GACATCTTCCCGAGGCTCCCTGAATGTGGCAGATG
AAAGAGACTAGTTCAACCCTGACCTGAGGGGAAAG
CCTTTGTGAAGGGTCAGGAG
```

Left: CpG sites at 1/10 nucleotides, constituting a CpG island. The sample is of a gene-promoter, the highlighted ATG consitutes the start codon.
Right: CpG sites present at every 1/100 nucleotides, consiting a more normal example of the genome, or a region of the genome that is commonly methylated.

Viterbi algorithm: **Notation**

Step i	i=0	i=1		i		i=L
Observation X:		x_1	...	x_i	...	x_l

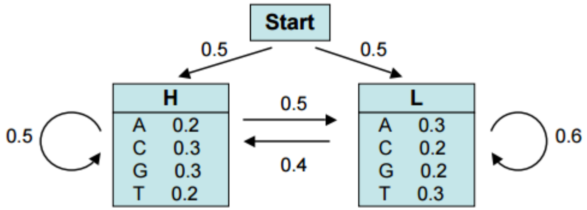
x_i : Observation at step **i**

$a_{k,l}$: Probability of the transition from state k to l

$e_k(x_i)$: Probability to observe element x_i in state k

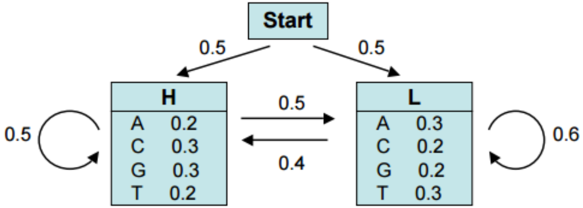
$V_k(i)$: Probability of the most probable path ending in state k at position i with observation x_i

Viterbi algorithm: **Example of Finding CpG Islands**



- Let's consider the following simple HMM.
- This model is composed of 2 states:
  - ✓ H (high GC content, e.g. coding DNA) and
  - ✓ L (low GC content, e.g non-coding DNA).
- The model can then be used to predict the region of coding DNA from a given sequence.

Viterbi algorithm: **Several Paths**



Consider the sequence S= **GGCACTGAA**

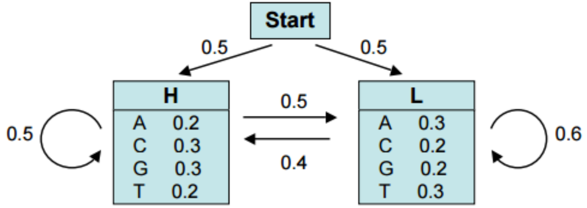
There are several paths through the hidden states (H and L) that lead to the given sequence.

Example: P = **LLHHHHLLL**

The probability of the HMM to produce sequence S through the path P is:

$$p = a_{0L} * e_L(G) * a_{LL} * e_L(G) * a_{LH} * e_H(C) * \ldots$$
$$p = 0.5 * 0.2 * 0.6 * 0.2 * 0.4 * 0.3 * \ldots$$
$$p = \ldots$$

Viterbi algorithm: *A dynamical Programming algorithm*



GGCACTGAA

There are several paths through the hidden states (H and L) that lead to the given sequence, but they do not have the same probability.

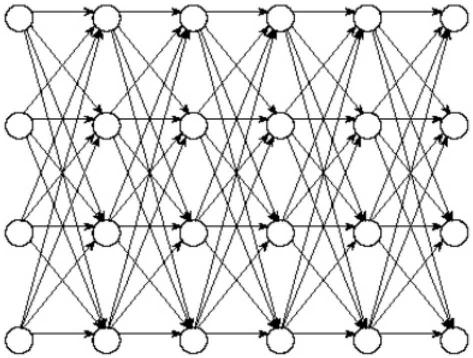
The **Viterbi algorithm** is a dynamical programming algorithm that allows us to compute $\boxed{\text{the most probable path}}$. Its principle is similar to the DP programs used to align 2 sequences (i.e. Needleman-Wunsch)

Source: Borodovsky & Ekisheva, 2006

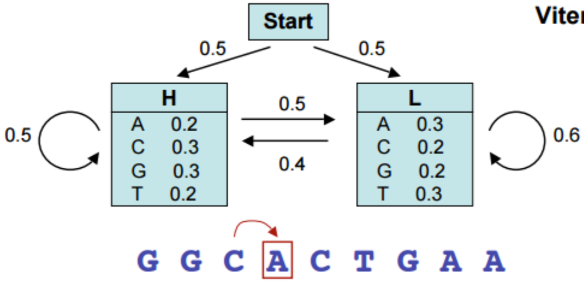
The edit graph for the decoding problem

Find the best parse of a sequence (the highest probability):

Search for the longest path:



Viterbi algorithm: **The most probable path** $V_k(i)$



Viterbi algorithm: principle

Suppose the probability $V_k(i)$ of the most probable path ending in state k with observation x_i is known for all states k, then:

$$V_l(i+1) = e_l(x_{i+1}) \max_k (V_k(i) * a_{kl})$$

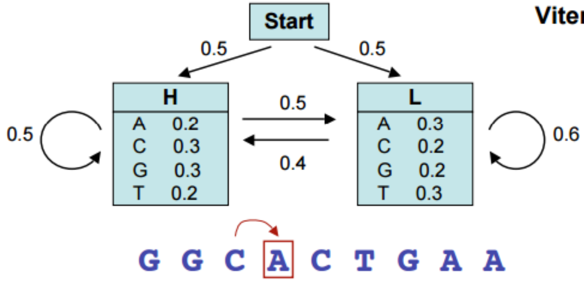
Probability of the most probable path at the end state *l*

Probability to observed element x_{i+1} in state *l*

Probability of the most probable path ending in state *k at* position *i*

Probability of the transition from state *k to state l*

Viterbi algorithm: *The probability of $V_H(4)$*



Viterbi algorithm: principle

The probability of the most probable path ending in state **k** with observation "i" is

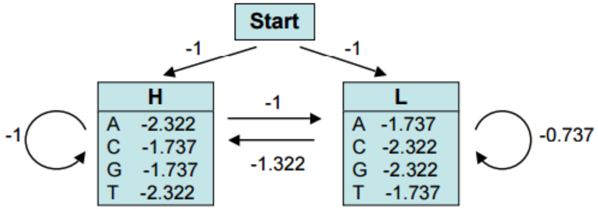
$$V_k(i) = e_k(x_i) \max_s (V_s(i-1) * a_{sk})$$

In our example, the probability of the most probable path ending in state **H** with observation "A" at the 4th position is:

$$V_H(4) = e_H(A) \max_s (V_L(3) * a_{LH}, V_H(3) * a_{HH})$$

We can thus compute recursively (from the first to the last element of our sequence) the probability of the most probable path.

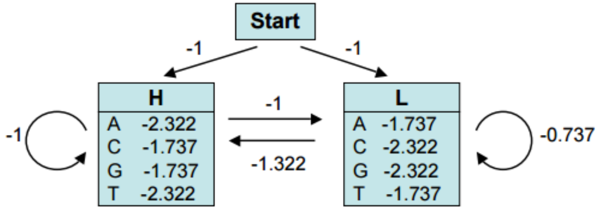
Viterbi algorithm: *Logs instead of Probabilities*



Remark:

- For the calculations, it is convenient to use the log of the probabilities (rather than the probabilities themselves).
- Indeed, this allows us to compute sums instead of products, which is more efficient and accurate.
- We used here $log_2(p)$

Viterbi algorithm: *Maximum at the first position*



GGCACTGAA

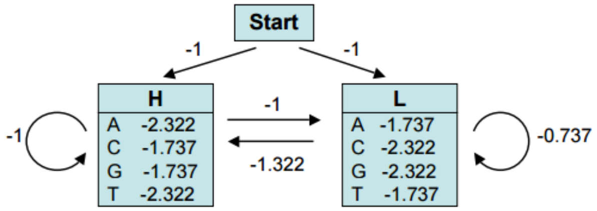
Probability (in $\log_2$) that G at the first position was emitted by state **H**

$$\boxed{V_H(1) = -1 - 1.737 = -2.737 \quad \text{(Maximum)}}$$

Probability (in $\log_2$) that G at the first position was emitted by state **L**

$$V_L(1) = -1 - 2.322 = -3.322$$

Viterbi algorithm: *Maximum at the second position*



GGCACTGAA

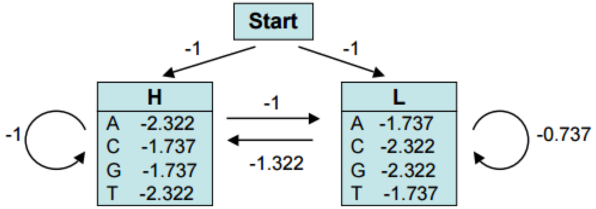
Probability (in $\log_2$) that G at the 2nd position was emitted by state **H**

$$\begin{aligned}V_H(2) &= -1.737 + max\left(V_H(1)+a_{HH}, V_L(1)+a_{LH}\right)\\ &= -1.737 + max\left(-2.737-1, -3.322-1.322\right)\\ &= -5.474 \text{ (obtained from } V_H(1) \text{)} \quad \text{(Maximum)}\end{aligned}$$

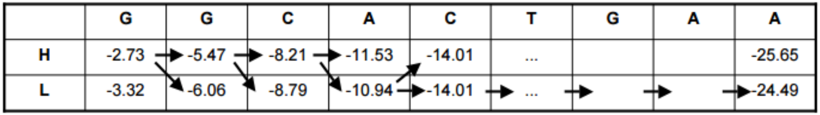
Probability (in $\log_2$) that G at the 2nd position was emitted by state **L**

$$\begin{aligned}V_L(2) &= -2.322 + max\left(V_H(1)+a_{HL}, V_L(1)+a_{¿}\right)\\ &= -2.322 + max\left(-2.737-1, -3.322-0.737\right)\\ &= -6.059 \text{ (obtained from } V_H(1) \text{)}\end{aligned}$$

Viterbi algorithm: *Compute Iteratively the Probabilities*



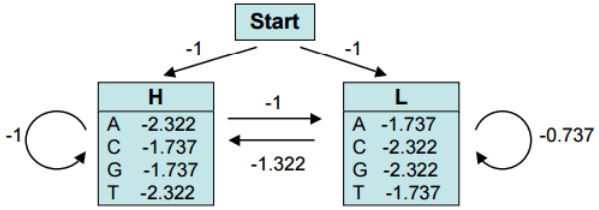
GGCACTGAA



	G	G	C	A	C	T	G	A	A
H	-2.73	-5.47	-8.21	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	-10.94	-14.01	...			-24.49

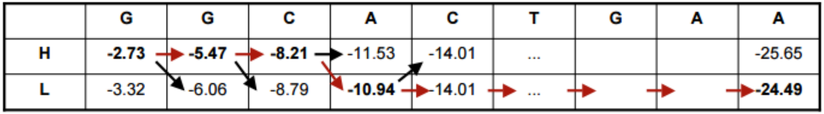
We then compute iteratively the probabilities $V_H(i)$ and $V_L(i)$ that nucleotide x_i at position i was emitted by state H or L, respectively. The highest probability obtained for the nucleotide at the last position is the probability of the most probable path. This path can be retrieved by back-tracking.

Viterbi algorithm: *Backtracking*



GGCACTGAA

back-tracking
(= finding the path which corresponds to the highest probability, -24.49)



	G	G	C	A	C	T	G	A	A
H	**-2.73**	**-5.47**	**-8.21**	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	**-10.94**	-14.01	...			**-24.49**

The most probable path is: **HHHLLLLLL**

Its probability is $2^{-24.49} = 4.25E\text{-}8$
(remember that we used $\log_2(p)$)

Viterbi algorithm: *Summary*

- The Viterbi algorithm is used to compute the most probable path (as well as its probability).
- It requires knowledge of the parameters of the HMM model and a particular output sequence ,and
- It finds the state sequence that is most likely to have generated that output sequence.
- It works by finding a maximum over all possible state sequences.

In sequence analysis, this method can be used for example to predict coding vs non-coding sequences.

The Viterbi algorithm

Input: $x = x_1,\ldots,x_N$

Initialization: $V_0(0) = 1$ (0 is the imaginary first position)

$V_k(0) = 0$, for all $k > 0$

Iteration: $V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$

$\mathrm{Ptr}_j(i) = \mathrm{argmax}_k\, a_{kj} V_k(i-1)$

Termination: $P(x, \pi^*) = \max_k V_k(N)$

Traceback: $\pi_N^* = \mathrm{argmax}_k\, V_k(N)$

$\pi_{i-1}^* = \mathrm{Ptr}_{\pi i}\,(i)$

Problem 2: Evaluation

Finding the probability a sequence is generated by the model

Forward algorithm: *Notation*

Step i	i=0	i=1		i		i=L
Observation X:		x_1	...	x_i	...	x_l

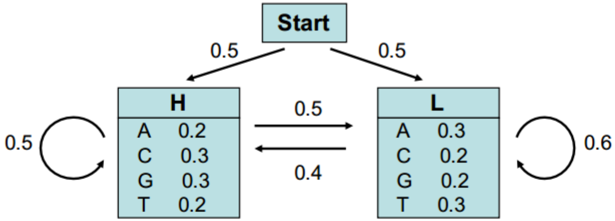
x_i : Observation at step **i**

$a_{k,l}$: Probability of the transition from state k to l

$e_k(x_i)$: Probability to observe element x_i in state k

$f_k(i)$: Probability of the observed sequence up to and including x_i and ending in state k

Forward Algorithm: **CpG Example**



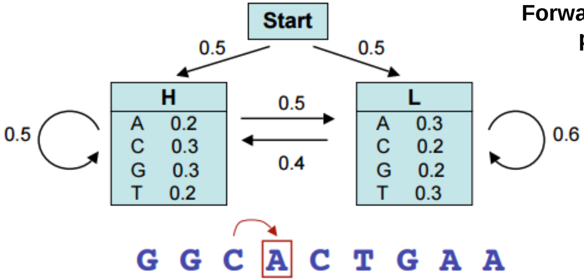
Consider now the sequence S= **GGCA**

What is the probability P(S) that this sequence S was generated by the HMM model?

This probability P(S) is given by the sum of the probabilities $p_i(S)$ of each possible path that produces this sequence.

The probability P(S) can be computed by dynamical programming using either the so-called **Forward** or the **Backward** algorithm.

Forward algorithm: $f_k(i)$, *Probabilty of x_i ending in state k*



Forward algorithm: principle

Suppose the probability $f_k(i)$ of the observed sequence ut to x_i, ending in state k

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) * a_{kl}$$

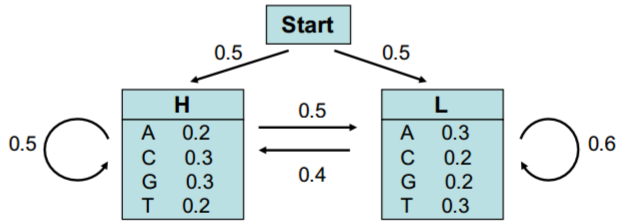
Probability to observed sequence up to x_{i+1} ending in state *l*

Probability to observed element x_i in state *l*

Probability to observed sequence up to x_i ending in state *k*

Probability of the transition from state *k to state l*

Forward Algorithm: **First Step:** $f_H(G)$ **and** $f_L(G)$

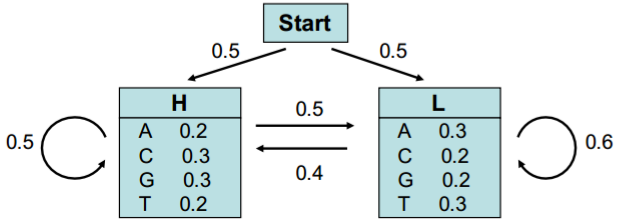


Forward algorithm

Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15			
L	0	0.5*0.2=0.1			

Forward Algorithm: **Second Step:** $f_H(G)$

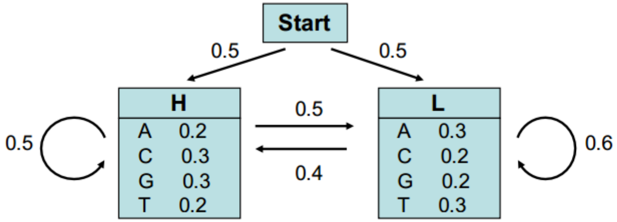


Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15 →	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345		
L	0	0.5*0.2=0.1 →			

S

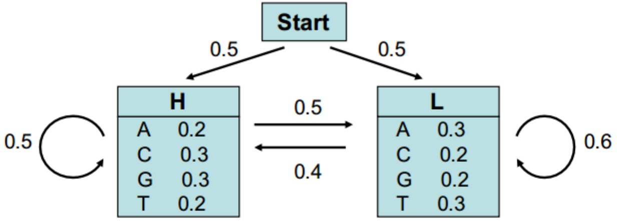
Forward Algorithm: *Second Step: $f_H(G)$ and $f_L(G)$*



Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345		
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027		

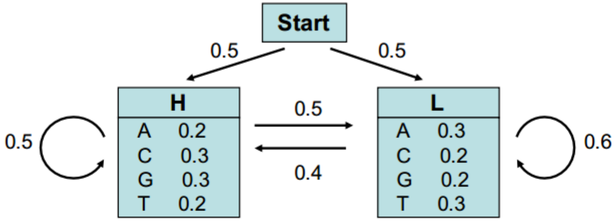
Forward Algorithm: *All steps: $f_H(i)$ and $f_L(i)$*



Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345	... + ...	
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027	... + ...	

Forward Algorithm: *Last Step: $f_H(L)$ and $f_L(L)$*



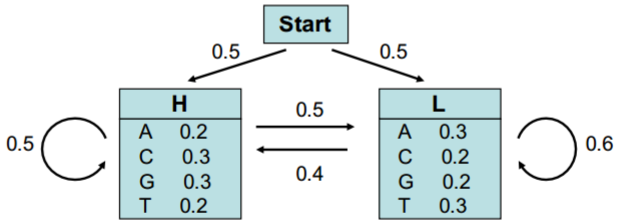
Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345	... + ...	0.0013767
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027	... + ...	0.0024665

$\Sigma = 0.0038432$

=> The probability that the sequence S was generated by the HMM model is thus P(S)=0.0038432.

Forward Algorithm: *Significance of the probabilty*



The probability that sequence S="GGCA" was generated by the HMM model is $P_{HMM}(S) = 0.0038432$.

To assess the significance of this value, we have to compare it to the probability that sequence S was generated by the background model (i.e. by chance).

Ex: If all nucleotides have the same probability, $p_{bg}=0.25$; the probability to observe S by chance is: $P_{bg}(S) = p_{bg}^4 = 0.25^4 = 0.00396$.

Thus, for this particular example, it is likely that the sequence S does not match the HMM model ($P_{bg} > P_{HMM}$).

Problem 3: Learning

Estimation of the HMM Parameters when state sequence is known

Estimation of the HMM Parameters when state sequence is known

Counting:

- When all the paths are known, we can count the number of times each particular transition or emision is used in the set of training sequences.
- Let be A_{kl} and $E_k(b)$:
  - ✓ $a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$
  - ✓ $e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$

Summary

The **Viterbi algorithm** is used to compute the most probable path (as well as its probability). It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

In sequence analysis, this method can be used for example to predict coding vs non-coding sequences.

In fact there are often many state sequences that can produce the same particular output sequence, but with different probabilities. It is possible to calculate the probability for the HMM model to generate that output sequence by doing the summation over all possible state sequences. This also can be done efficiently using the **Forward algorithm**, which is also a dynamical programming algorithm.

In sequence analysis, this method can be used for example to predict the probability that a particular DNA region match the HMM motif (i.e. was emitted by the HMM model).

Summary

Remarks

To create a HMM model (i.e. find the most likely set of state transition and output probabilities of each state), we need a set of (training) sequences, that does not need to be aligned.

No tractable algorithm is known for solving this problem exactly, but a local maximum likelihood can be derived efficiently using the **Baum-Welch algorithm** or the **Baldi-Chauvin algorithm**. The Baum-Welch algorithm is an example of a forward-backward algorithm, and is a special case of the Expectation-maximization algorithm.

For more details: see Durbin *et al* (1998)